Live Repetition Counting

Ofir Levy Lior Wolf
The Blavatnik School of Computer Science
Tel Aviv University

Abstract

The task of counting the number of repetitions of approximately the same action in an input video sequence is addressed. The proposed method runs online and not on the complete pre-captured video. It analyzes sequentially blocks of 20 non-consecutive frames. The cycle length within each block is evaluated using a convolutional neural network and the information is then integrated over time. The entropy of the network's predictions is used in order to automatically start and stop the repetition counter and to select the appropriate time scale. Coupled with a region of interest detection mechanism, the method is robust enough to handle real world videos, even when the camera is moving. A unique property of our method is that it is shown to successfully train on entirely unrealistic data created by synthesizing moving random patches.

1. Introduction

Given an input video capturing a scene in which the same action is repeated multiple times in consecutive cycles, we seek to count the number of repetitions. The video can depict a bird flapping its wings, a hand playing a guitar, a person performing multiple repetitions of the same exercise etc. We place no restrictions on the nature of the performed action, and are able to deal with camera motion.

Our method does not assume that the video is segmented in time. It automatically detects the start and end points of the sequence of repetitive actions, and counts the repetitions on a live video stream: the repetitive nature is identified early on and counts commence and update automatically. Despite working online, counts are not lost during the time that the method detects the repetitive nature of the video.

Our method successfully handles live real world video inputs as well as video collected from youtube and other sources. There are significant challenges. In many videos, there are only a handful of repetitions; the cycle length changes significantly throughout the video and each repetition is often visually different from the other repetitions.

We propose a novel online scheme that employs a shifting window in time. Every window is examined and the number of repetitions is estimated. We pose this as a multi-class classification problem. The information is then integrated smoothly across multiple shifting windows to produce the ongoing counts.

The underlying classification mechanism employs a Convolutional Neural Network (CNN). Since the problem is well defined regardless of the content of the video, we are able to train the network on a completely unrealistic synthetic video. This is the first work that we are aware of, which is able to build a real world system based entirely on such data (previous successful usage of synthetic data relied on *realistic* computer graphics data). This is noteworthy: the generalization capability of CNNs is high enough to enable a new venue of defining computer vision tasks by creating synthetic data that does not resemble any real world signal. Moreover, the probability estimation is valid enough to allow for the entropy of the network's output to be used as a score for detecting the start and end of the action and the appropriate time scale.

The CNN is the only learned part in our system. No real images or videos were used for training. There are a few time scale parameters and thresholds that are set once globally based on observing a few videos. The method is robust to the choice of these parameters. Not training on real world data, makes the resulting system extremely general and it can be used without any modifications on a variety of datasets. The entire implementation runs at a frame rate of 50 frames per second on live video on a conventional PC.

1.1. Previous work

The task that we solve is well defined at the semantic level. Moreover, it seems that humans are able to perform this task relatively easily (depending on the input) and that the task has applications in high throughput biological experiments, activity monitoring, sports, and gaming. It is, therefore, not surprising that the problem has already gained some attention. However, our method is very different from all previous work. While most previous approaches were based on frequency domain analysis or on matching, our method is based on a novel approach of cycle length esti-

mation, turning the counting problem on its head. Moreover, (i) the only other work we are aware of that is able to work on live input, and not as post-processing applied to the entire video, are in the domain of motion capture (Mocap) sequence analysis; (ii) the problem of finding where the repetitions start and end is mostly neglected. Lastly, carefully examining the literature, no previous work was evaluated on truly unconstrained video inputs, and the vast majority were evaluated on a handful of samples that were very restricted. Our ability to perform well on a variety of real world 2D videos including a live video feed of casual users also sets us apart from all previous work that we are aware of.

Spectral or frequency domain methods such as Fourier transform based methods or wavelet analysis methods dominate the current literature [3, 5, 35, 29]. A limitation of these methods is that it is assumed that the action frequency would emerge as a discernible peak at a time frequency graph of one type or another. Except maybe for video sequences of simple motions or very uniform repetitions (cf. [34], Fig. 2, or Tab. 2 of [29]), the amount of variation in appearance between repetitions and the variation in action length means that no clear peak is visible. Similar to our method, a time scale or other resolution parameter is either implicitly or explicitly used, which limits the range of action lengths that can be detected. In our method, we overcome this limitation by automatically performing live selection between multiple detectors. In the previous literature, time scale selection is mostly ignored. In addition, the frequency domain methods in the literature work on the entire video as post processing and are not supplied with a proper mechanism to identify the start and end points of the sequence of repeated actions. In our experiments, we make a comparison to [3] and show that even using the best value for its parameter, selected in hindsight, separately for each video, there is a sizable performance gap in our favor.

Matching can also be used for counting. In [21] the geometric constraints arising from multiple repetitions of the same motion as the viewpoint changes are used to detect and segment repeated motions. While this may lay foundations for a counting system, counting is not performed or evaluated. 3D reconstruction using repetitions is also the focus of [31, 32], with applications such as gait recognition.

Autocorrelation is employed in [1, 40] and applied to a handful of samples in very restricted domains. Recently, two autocorrelation systems were developed based on matching visual descriptors [30, 23]. While both systems display a screen shot of a counting application, they are both post processing methods and are only applied in specific domains on constrained videos. String matching is used in [12] and preliminary results are shown.

One contribution that might be suitable for counting is the hierarchical motion clustering method of [42]. Given the entire video, frames are grouped to form segments of the same class of motion. This could be employed to identify repeated segments of the same type. However, this is still hypothetical, requires seeing the entire video before-hand, and the system of [42] is yet to be applied to unconstrained video. Most of the methods that have been suggested exclusively for the segmentation of motion capture (Mocap) sequences, such as [24, 2], share the same limitations.

An online method for performing hierarchical segmentation of Mocap and 2D sequences is presented in [9]. Similar to ours, it employs a sliding window in time, works online, and is able to identify the borders of the repeated actions. The experiments that focused on 2D data were based on silhouettes extracted using background subtraction, thus the method is not robust to camera motion. Moreover, the 2D experiments apply a flat segmentation hierarchy and focus on detecting the change point between different actions such as walking, jumping, and boxing, and not on counting.

In the case that the repeated motion is from a known class and is the type of motion that has clear boundaries in time, action detection [26, 15] can be employed. A direct counting of the detected events would then provide the desired output. While this is limited to specific cases, it has the advantage of possibly performing better in the presence of distracting motions that are also present in the scene.

Deep learning We use a deep neural net [14, 20] in order to solve a very specific classification problem – given a short video sequence, predict the cycle length of the repeated motion that it captures. We employ, as the very first layer, a 3D convolution. Such convolutions are natural in video processing and have been used for video action recognition [18]. Previous work on employing deep architectures in action recognition include [22, 19, 37].

Use of synthetic data in visual perception Our work employs synthetic data for training. It is unique in that the synthetic data is unrealistic (see Sec. 2.1). However, training computer vision systems by using realistic or semi-realistic synthetic data is a common practice. Matikainen et al. [25] generated synthetic action sequences from motion capture data, using a computer graphics human model. Chen and Grauman [4] were able to create synthetic clips of human action by integrating labeled still images with unlabeled human action video. In [28], computer graphic human figures where placed on background images to create a training set for pedestrian detection. In pose estimation, realistic computer graphics datasets are very successfully used in order to train pose recognition systems [36, 11]. In another domain, state of the art results in text detection were achieved using a CNN trained on synthetic text examples [16].

2. The proposed approach

Our design combines a learned classifier encompassed by a larger system. Indeed, counting does not call for an end to end trainable system: cycle lengths can be estimated

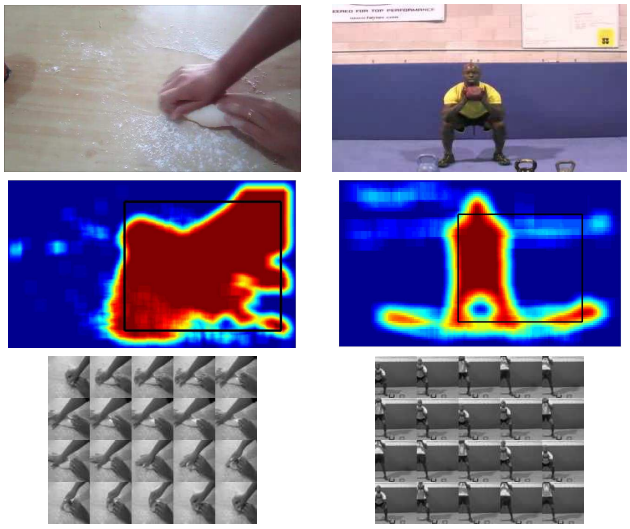

Figure 1. Example sequences. (first row) A frame from the video. (second row) The heat map of the matrix E_2 (before thresholding) used in order to detect the ROI and the estimated ROI. (third row) The 20 non sequential frames as passed to the CNN classifier.

based on observations, while integration in time is well defined mathematically, and there no need to learn it. For clarity of exposition, we break down our method into two components: the core system that is used to perform counting and the outer system that decides when to start and stop counting and selects between multiple time scales.

The core system analyses blocks of 20 frames at a time. The ROI is detected automatically for each block independently of the other video blocks as described in Sec. 2.3. The ROI is then resized to a size of 50×50 pixels resulting in video blocks of size $50 \times 50 \times 20$ as depicted in Fig. 1. This block is then passed to a CNN, which estimates the cycle length in this video block, see Sec. 2.2. As mentioned, this classifier is trained on a set containing tens of thousands of synthetic examples. These are generated at random using one of four motion patterns, each with its associated set of random parameters, see Sec. 2.1. The classification output is integrated over time as described in Sec. 2.4.

The outer system is presented in Sec. 3. The core system above is applied to the video at multiple time scales, simply by sampling the input frames in time. This enables a wide coverage of repetition lengths. The selection among the time scales is done by examining the entropy of the probabilities assigned by the underlying CNNs. The lower the entropy, the higher the confidence we have in the result of the CNN classification. The same entropy is also used to determine where to start and stop the counting. A simple state machine based on this entropy detects that a repetitive action has possibly started, and if the entropy remains low for a set number of seconds, it performs counting. The count-end event is detected similarly, and following a correction, if necessary, to the last count, the system is reset and is able to start a new count.

Note that we treat the problem of predicting, per video block, the cycle length as a classification problem and not as a regression problem. There are three very good reasons for this choice. First, time filters that capture repetitions every l frames are very different from those meant to capture repetitions every $l+1$ or $l-1$ frames. This plays in our favor – the various discrete states are well distinguishable. In our experiments, we provide an empirical comparison to a very similar CNN, where the output layer is optimized to perform regression, showing that classification considerably outperforms regression. Second, the extensive use we make of entropy in our live method, both to select the appropriate time scale and to detect the boundaries of the action, is only possible within a multiclass classification problem. Finally, in Deep Learning, it is recommended to strongly prefer classification over regression, whenever discretization is possible [7].

The classification approach is not without limitations. The cycle lengths in the data itself are not integers, but in practice the method seems to be robust to such rounding effects, since each cycle is captured by multiple sliding windows of the same detector that are all integrated in time. Moreover, since we use multiple detectors, the same range of cycle lengths (in seconds) is often captured by multiple detectors that quantize the cycle length at multiple resolutions. Another challenge, shared by both the classification and the regression approaches, is that a cycle length of l also defines cycle lengths of nl, where n is an integer. In our experience, the most rapid cycle length is the one that often gets counted. The reason is that shorter cycles are supported by more evidence and are, therefore, typically associated with higher probabilities.

2.1. Synthesizing data

The synthetic data used for training follows four patterns, all displaying squares of a low frequency pattern moving in front of a white noise background. Three to five such squares are present in every frame, all with random sizes, and all move following the same motion pattern but with different random motion parameters. The labels denote the cycle length, which is fixed for each created synthetic sample. The first pattern is of linear motion along a path of random length with a jump back to the starting location after the cycle length number of frames. The second pattern is of a circular motion along a path ending at the starting position every cycle length frames; the radius is set at random. The third pattern is of expanding and shrinking in a cyclic manner independently in both axes. The fourth and last pattern displays in place rotation up to a certain degree and back to the starting rotation at the end of the cycle. To add variability, we also create sequences in which the patterns are randomly mixed within the same sequence.

The location of the center of each square is shifted in-

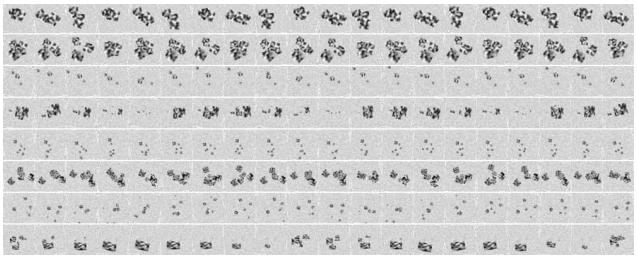

Figure 2. Samples of synthetic sequences for cycle lengths ranging from 3 (top row) to 10 (bottom row). Each row is a sequence from one of the 4 types used to synthesize data. As can be seen, there is a large variability in the size of the moving squares used.

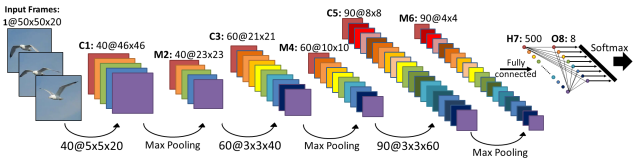

Figure 3. Outline of the architecture we have used in this work, which consists of interleaving convolution/maximization layers followed by a fully-connected hidden layer.

dependently at each frame from the predetermined cyclic path, by uniformly and independently sampling the amount of shift in a fixed range. In addition, a global rotation of the frame, taken uniformly from - 20° up to 20° , independently to each frame, is applied to half of the sequences. This simulates global camera motion.

All synthetic sequences contain 20 frames of size 50×50. Cycle lengths in the range of 3..10 are used. A cycle length of 2 is too short, and a cycle length of more than 10 does not fully repeat in 20 frames. Examples of sequences are shown in Fig. 2.

In the training process, 30,000 random training sequences are used, as well as 5,000 validation sequences. These numbers are within the range that appears in the computer vision literature for CNNs. While object recognition systems are often trained on ImageNet's [6] million images, other systems, e.g., [38, 39] use a comparable number.

2.2. CNN architecture and training

We train our CNN to classify the cycle length within a fixed number of frames. The overall architecture is shown in Fig. 3. The input is a 3D tensor of size 50 by 50 by 20 capturing a fixed ROI taken across 20 (non-consecutive) frames and resized to 50 by 50 pixels. This is denoted by 1@50x50x20, since we have a single channel gray image.

The input is given to a convolutional layer (C1) with 40 filters of size 5x5x20, which encodes temporal patterns across the entire 20 frames. Max pooling is then performed, followed by interleaving convolution/maximization layers (C3-M6), resulting in an output with spatial resolution of 4x4. This low resolution makes sense since the phenomenon we aim to capture is global across the entire frame. The size of the hidden layer H7 is taken to be 500, which seems to be sufficient to represent the space of motion repetitions for limited sized blocks. The output size contains one neuron per 8 possible classifications corresponding to the 8 cycle lengths from 3 to 10.

After each convolution-maximization pair, and after the hidden layer H7 and the output layer O8, a learned activation bias (one per channel) is added and the Rectified Linear Unit (ReLU) [8] is then applied. The output of the last layer is fed to a softmax function, and the cross-entropy loss is used. For the alternative regression system presented in our experiments, the output layer O8 has a single neuron representing the cycle length and the Euclidean loss is used.

2.3. Detecting the ROI

In many videos, the relevant motion occurs in a relatively small part of the image. Therefore, given a block of 20 non-consecutive frames, we compute a rectangular ROI that is aimed at capturing the majority of the observed motion.

Let the tensor T hold the pixels of a single 20 frame video block of size WxHx20. The first step is to compute a 2D map containing the standard deviation along the third dimension of T. This map, thresholded at its mean, produces a binary map E. The map E is convolved with a square all-ones kernel of size 10. A second threshold is applied at the value of 80 to produce a second binary map E_2. It contains all pixels for which 80% of the 10×10 neighborhoods in E have a value of +1. The ROI is computed separately to the two axes. Let $\{i_k, j_k\}_k$ be the set of coordinates of all +1 values in E_2. The ROI boundaries along the x-axis (y-axis) at the 3rd and 97th percentiles of $\{i_k\}$ ($\{j_k\}$).

This simple method, shown in Fig. 1, was found to be effective enough for a wide range of video sequences. The second example of this figure depicts a case in which a human would mark the bounding box differently. The automatically selected ROI contains the moving shadow. While specialized modules can be added to deal with such artifacts, we opt for a simple real-time solution, pointing to the robustness of our overall design. In addition, when the camera is moving, the assumption behind the ROI computation is violated and the ROI becomes the entire image sans a thin margin. As demonstrated experimentally, the other components of the counting method work well even in such cases. Lastly, note that action ROIs have gained recent attention [17, 27]. However, these contributions are off-line (require the entire video), while our method is live.

2.4. Integration of counts

While the method employs multiple detectors at multiple time scales, for clarity, we begin our exposition of the counting integration method with a single detector, which is associated with a temporal subsampling parameter N. The detector collects blocks of 20 frames, sampled uniformly

from a video segment of length $20N$ frames. After the computation of the ROI, the CNN classifier is applied to the video block. This process of collecting 20 frames and producing a label is repeated every N video frames, to match the gap between the sampled frames, i.e., the second sampled frame from the first block becomes the first frame in the second block. The labels from the consecutive blocks are integrated to produce an online counter.

Every video block (20 frames) produces a label $\hat{y}$ and the associated probability $p_{\hat{y}}$ computed by the softmax function (Sec. 2.2). The integration module is best illustrated by an example, see Fig. 4. This module works online and holds two counters that are updated after each video block is analyzed: R, which is the current repetition count and holds the estimated number of repetitions from the motion's start, and C which is a counter that holds the number of frame strides since the last update of R.

After the first video block in which motion is identified, a few repetitions, each of length $\hat{y}$ have occurred already. The counter R is therefore set to $\lfloor 20/\hat{y} \rfloor$, and C is set to $20 \bmod \hat{y} = 20 - \hat{y} * R$. After the analysis of the next block, which produces a new label $\hat{y}$, the frame counter is advanced $C = C + 1$, and R would remain unchanged unless $C \geq \hat{y}$. In this case the video has advanced enough to complete an additional repetition of length $\hat{y}$, and therefore the repetition counter is incremented $R = R + 1$ and the counter of frame strides is reset to $C = 0$. This process is repeated for every subsequent video block.

3. Working in multiple time scales and handling intro and outro video segments

At every time point, the method inspects 20 frames that are sampled uniformly every N frames. The parameter N should be set in accordance with the expected frequency of motion. The CNN classifier outputs cycle lengths between 3 and 10 frames, which translate to a repeated action of a duration ranging from $3N/30$ seconds to $10N/30$ seconds, assuming 30 frames per second.

We suggest employing multiple detectors working at different time scales. In our experiments, we set the multiple values of N to be 2, 5, and 8. This range of possibilities was set to capture the rapid, moderate, and relatively slow actions we observe in our data. Naturally, more detectors could be added in order to provide a wider coverage.

Note that since the low range of one detector can overlap the high range of another, the same cycle length in seconds is observed by multiple detectors. For example, for the three detectors we employ, the rapid and the moderate detectors share the range 0.5–0.66 seconds, while the moderate and slow detectors share the range of 0.8-1.66 seconds. Overall, almost half of a unified range of 0.2–2.33 seconds per cycle is covered by more than one detector.

Each detector provides for each video frame a vector of probabilities derived from the softmax layer $p_i, i = 3..10$. During training, these probabilities were optimized to minimize the cross-entropy loss and, therefore, would maximize the log probability of the correct class. We discovered that the entropy of the predictions $H(p) = -\sum_i p_i \log p_i$ is a useful uncertainty score. When it is low, the detector is much more likely to provide the valid cycle length. We, therefore, use the entropy extensively in our method in order to decide if a repeated action is taking place and at which time scale. Note that this is the first usage, that we know of, of the entropy of classifier predictions as a direct signal in a live working system (entropy maximization is often used in active learning, e.g., [13], however this is a different mechanism). This is only made possible through the high accuracy of our underlying classifier.

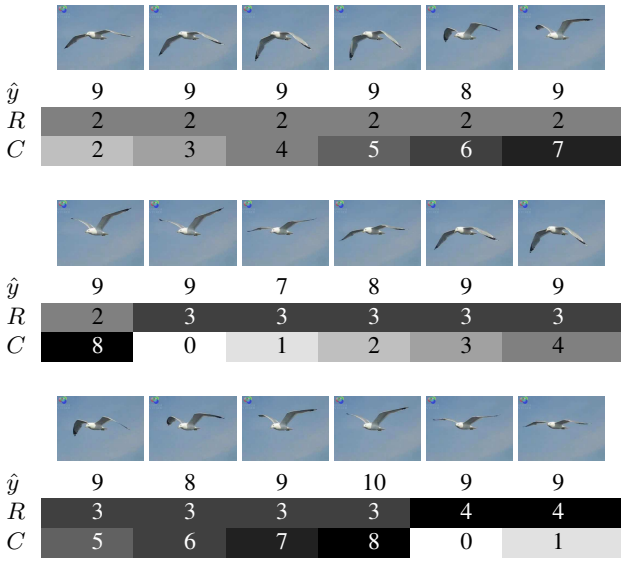

Figure 4. An example of the count integration process for a single detector. Shown are the frames from the beginning of each video block, the current estimation of the cycle length $\hat{y}$, the number of repetitions seen so far R, and the variable C that holds the numbers of frame strides since the last update of R. More details can be found in Sec. 2.4. This example shows a relatively clean case in which there was little variation in the cycle length. In the first video block, $\hat{y}$ is estimated to be 9, therefore R is set to 2 and the register C is set to $20-2*9=2$. The method then continues from one video block to the next. A count is added to R when $C \geq \hat{y}$.

Off-line selection of the time scale In the experiments for which one detector is selected per video, the analysis is done offline for the entire video. The detector that gives the lowest average entropy is selected. Counting for each detector is performed independently as described in Sec. 2.4.

Live selection and integration of multiple time scales To perform live counting with three concurrent detectors, we hold a global counter, in addition to the registers R and C we hold per detector. The three detectors we employ provide estimations every 2, 5, or 8 frames. Every 40 frames all three provide estimations at once and we perform synchro-

nization. The average entropy of the detectors is compared: for the rapid counter 20 measurements are averaged, while for the other two only 8 or 5 exist.

The detector with the lowest entropy is selected and the count difference in this counter from the reading 40 frames ago is added to the global counter. The individual counters R of the three detectors are zeroed, and the frame counters C of the two detectors that were not selected are updated such that the time length since the last repetition is similar. For example, if the moderately rapid detector was elected and the value of its C register is 6, this translates to a new value of C of $6 * 5/2 = 15$ for the rapid detector, and only $6*5/8 \approx 4$ for the slow detector. Since these updated values possibly contain more than one repetition at the currently estimated cycle length, the C is taken modulo the currently estimated cycle length. If, for example, the currently estimated cycle length ($\hat{y}$) of the rapid detector is 6, then the new value of C is updated to be $15 \bmod 6 = 3$.

In order to keep the counting live in between synchronization points, the last selected detector updates a separate live counter shown to the users. At the synchronization point, when the selection of the lowest entropy detector in the last 40 frames is performed, the live counter is updated to match the global counter. This very rarely changes the value of the live counter.

Repeated action start and end A moving average over the last 7 entropy readings is kept for each detector, and the minimal value of the three smoothed entropies is considered. An entry threshold of 1.04 on this value (half the random entropy) is used to alert of the possibility of repetitions. As can be seen in Fig. 5, this threshold separates well the 20-frame blocks of repetitive actions vs. non repetitive blocks. Once the entry threshold is crossed, the user is alerted that a new possible repetitive action is taking place ("new hypothesis"), and the current count is displayed. Since sometimes there is a transient drop in entropy, if within the first 4 repetitions or 4 seconds (whichever is first) the smoothed entropy rises above that threshold, the count is terminated. If the entropy stays below the threshold, the user is presented with a permanent counter.

The minimal smoothed entropy among the three detectors is also used to stop the counter after the first 4 seconds with the same threshold of 1.04. Since the entropy is smoothed in time, the counting will sometimes stop with a slight delay. To fix this, we examine the history of the relevant unsmoothed entropy (last 7 readings), and detect the point of maximal change in entropy. The count at this specific point in time is then taken as the final count. This means that sometimes, at the end of the count, the counter rewinds by one count, which is not entirely desirable. An alternative would be to display counting with a slight delay.

Once the counting stops, the method waits for a new counting event. The entire process is completely automatic and the implementation runs live for hours at a time.

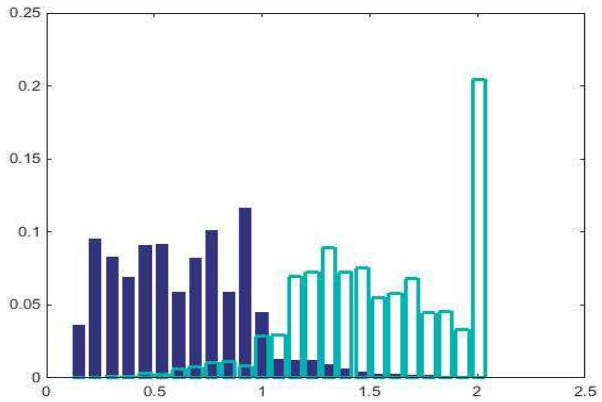

Figure 5. Histograms of the smoothed entropy of 20-frame blocks at time points that are within repetitive actions (solid blue), and outside repetitive actions (outlined magenta) obtained from the long continuous video of Sec. 4. As can be seen the threshold at 1/2 the random entropy (1.04) separates well between the groups. Repetitive blocks above this threshold are predominantly taken from the time points at the start of the action; non repetitive blocks below the threshold may create a transient hypothesis but usually do not amount to a false detection.

4. Experiments

We provide experimental results covering the various aspects of the method. The evaluation is performed on very diverse sets of challenging real world videos, containing camera and background motion and depicting complex actions that differ in length and in appearance between repetitions. Note again that the real videos are used only for testing and do not take part in the training process. The exact same implementation is used throughout the experiments using the same parameters, unless otherwise noted.

Training on the synthetic data Training stopped, via an automatic early stopping mechanism when the validation error stopped decreasing at a value of 8.5% at epoch 102. Each epoch took 3 minutes on a NVIDIA GTX580 GPU. This error is per video block and not directly comparable the counting error reported below.

Previous benchmakrs We approached authors of all relevant papers and collected 6 previous benchmarks [33, 30, 3, 23, 10, 41]. As can be seen in Tab. 1, our method succeeds on all existing benchmarks, surpassing existing results. This is despite the fact that our method solves a harder problem (online processing vs. offline batch processing). It is also robust enough to handle the videos provided by [23], which are binary foreground masks, without modifications.

The YouTube benchmark For the purpose of benchmarking, we collected a dataset of 100 videos containing repetitions, the vast majority of which are from youtube. This test dataset displays a good mix of domains, including exercising, cooking, building, living creatures, etc. In order to create a clean benchmark from a very diverse set of videos,

	Entire youtube benchmark (100 videos)				Moving	Youtube benchmark in reverse			Mov. rev
method	Full	no ROI	regression	[3]	Full	Full	no ROI	regression	Full
manual selection	3.17±0.8	3.28±0.9	10.81±1.7	29.80±4.6	3.13±0.8	3.52±1.1	3.89±1.2	11.47±1.7	3.94±0.7
offline selection	6.18±1.4	8.58±2.1	N/A	N/A	6.58±1.4	6.97±1.3	8.16±1.9	N/A	5.83±1.0
online selection	6.84±1.6	9.72±2.2	N/A	N/A	7.35±1.1	7.36±1.8	8.77±2.3	N/A	7.04±1.0
detector median	21.2±3.4	22.6±3.5	28.84±2.9	N/A	19.1±3.5	22.4±3.9	23.2±4.1	31.36±3.4	23.9±4.0

Table 2. The error rates on the youtube benchmarks for our method (and 9 simplified variants) and a baseline method. Mean±Standard Error of the absolute error in percents are shown, i.e., the statistics of $\frac{|G-R|}{G} * 100$, where G is the ground truth count for a specific video, and R is the estimated count. The first columns correspond to the full system. Two additional types of variants are evaluated, one with the entire frame as ROI, and one where multiclass classification is replaced by regression. Results are shown for the original benchmark and the one played in reverse. Results are also shown for the moving camera subset. The method of [3] is used with the best possible parameter chosen per video based on the comparison to the ground truth. It cannot handle a moving camera and as a frequency based offline method, it is not affected by the playback direction. The rows correspond to manual selection of the best time scale, an offline selection of the time scale based on the average detector entropy, a dynamic online selection of the time scale, and just taking the median of the three detectors. The regression based system does not allow for entropy based selection among the detectors.

Source	#videos	baseline	Our
[33]	2	0	0
[30]	3	0	0
[3]	6	0.45%	0
[23]	50	4.95%	2.67%
[23] ([10]'s videos)	9	2.47%	0
[41]	6	N/A	4.46%

Table 1. A comparison of our method's performance on literature benchmarks. Reported are the absolute errors in percent. Our method matches or outperforms existing approaches even though it is the only method that processes the input online. The construction dataset of [41] was not used to evaluate counting before. Such results did not appear in [10] either. The two datasets of [23] contain only binary foreground masks. Our method handles those without any modification.

the videos are presegmented to include only the repeated action. We manually marked the individual repetitions and discovered that the average value of the ratio of the difference between the longest repetition and the shortest repetition over the mean repetition length is 31%, thus the videos display a very large variability in repetition length.

We perform extensive evaluations using the benchmark. As the main metric, we use the fraction in percent of the absolute difference in counts between the ground truth G and the estimated count R over the ground truth: $100\frac{|G-R|}{G}$ (absolute differences are easier to interpret than MSE). An upper bound on the performance of the method of [3] is also reported, where the original code is used and the best pseudospectrum parameter is selected for each movie separately such that the test error is minimized (authors of other publications were unable to share their software or to run it on our benchmark). To evaluate the effect of camera motion, we report separately results on a subset of the 100 videos that contains 30 videos of considerable camera motion.

In order to analyze the method's individual components, we evaluate our system without the ROI detection mechanism, and evaluate a system in which classification is replaced by regression. The scale selection is studied by evaluating multiple variants of the method (where appropriate) for the manually selected detector, the offline selection of detector based on mean entropy in the video (the system is online, only the selection is done offline), and for the online selection of detector, see Sec. 3; Also shown is a "no selection" option of taking the median out of the three counters.

The results are reported in Tab. 2. Overall, the counting process is accurate with an average error of around 7% for the online automatic selection of the time scale. Naturally, when the selection is done offline or given by an oracle, accuracy further improves. To put these results in context, when using a fixed values of $N = 5$ for all videos, i.e., without selection at all, the error is much higher: 21.31% for the full system on the entire benchmark, 22.73% on the reversed benchmark. The error rate is considerably higher for the two other values of N employed in our implementation $N = 2$ and $N = 8$.

It is also evident from Tab. 2 that the automatic ROI selection is beneficial. Classification greatly outperforms regression, as expected, see Sec. 2. Note that automatic time scale using entropy cannot be evaluated for the regression scheme. The baseline literature method, despite using hindsight-based selection of its parameter does not perform well on our benchmark. The subset of videos for which there is a significant camera motion does not seem to be more challenging than the entire benchmark.

Our method is online and therefore perceives the movie completely differently when played in reverse. In order to demonstrate its robustness, we also present results on the youtube videos played backward. As can be seen, the results are similar. Paired t-tests done for each of the settings separately, relieved that one cannot reject the null hypothesis that the mean performance of the two directions is the same at a confidence level of 0.05.

Most of the Youtube repetitive actions collected are tightly segmented (e.g., by shot cuts) at the source. However, a subset of 25 videos have non-repetitive segments before and after the repetitive action. This subset called youtube in-out (YTIO) was used to evaluate the perfor-

	YTIO	YTIO(rev)	live
true positive (found actions)	25/25	25/25	43/44
false positive	2/0	3/0	3/0
false negative (miss)	0/25	0/25	1/44
start time within 0.5 sec	20/25	20/25	38/43
start time within 1 sec	21/25	21/25	41/43
end time withing 0.5 sec	19/25	19/25	33/43
end time withing 1 sec	22/25	21/25	38/43
rewound with positive outcome	8/25	5/25	6/43
rewound with negative outcome	2/25	2/25	1/43
perfect counting	15/25	16/25	24/44
counting within 1 count error	18/25	19/25	35/44
counting within 10% error	22/25	22/25	36/44

Table 3. Statistics on videos that contain non-repetitive segments as well as repetitive ones. Shown are the detection statistics, the accuracy of the start and end signal, and the effect of the rewinding mechanism. Results are shown for the youtube in-out (YTIO) subset, for YTIO played backward, and for the annotated live video.

mance of the live detection method. The results are reported in Tab. 3. As can be seen, our method detects the true action in all videos, and has only 2 false positives (3 when played in reverse). The start and end time are approximately correct most of the time, and the rewinding mechanism reacts up to 40% of the time, and in the vast majority of these cases improves the outcome.

Benchmarking live performance The software was presented in the hall outside a large public event, where hundreds of users were free to interact with the system with minimal or no guidance. The overall feedback was extremely positive and the system seemed to count well almost the entire repertoire of actions performed by the users. Two **limitations** of the current implementation were identified: first, the implementation was not designed to deal with multiple persons entering the frame; second, some users tried very minute actions, which were missed.

In order to evaluate the live performance of the method on a continuous video, we collected an hour long uninterrupted video and annotated it. The results are presented in Tab. 3. Overall, out of 44 actions performed, only one was missed. The counting of these 43 actions was performed with an average absolute error of 6.2%. There has been 7 retractions of the last count, out of which 6 improved the outcome. The action started very closely to the actual start in the vast majority of cases. Note that the live video does not display the starting point since until repetitions take place, the repeated action cannot be detected. What is presented to the user is the new hypothesis message, followed by a counter message 4 repetitions or 4 seconds later. The shift in the start time is, therefore, estimated in retrospect. The absolute shift in the end point was around 0.3 seconds on average. Within the video, there were many non-repetitive motions and the system made 3 false positive counts, all in a range of less than 5 repetitions.

Sensitivity to parameters Similar to all other computer vision systems, the system has some parameters. As seen above, the same set of parameters works on literature benchmarks, youtube videos, and on uninterrupted live video without any modification. Many of the parameters' values are natural, e.g., time scale. Additional experiments were conducted in order to demonstrate that the system is robust with respect to its parameters. Specifically, the performance is highly stable to the parameters of the ROI, and the parameters could be tuned to get a slight improvement in performance over what is reported in Tab. 2. We made a deliberate choice not to train on our benchmarks and therefore such tuning was not performed. The system also presents a mild tradeoff between false positives and misses when altering the parameters that control the shift from a hypothesis into a permanent counter.

5. Summary

Identifying repetitions in video is harder than it seems at first glance: The motion of interest can vary in appearance and be limited to a portion of the frame and the cycle length can vary dynamically. In this work, we propose a CNN based online method that counts repetitions robustly. We present multiple technical novelties: the online design that uses a sliding classifier for counting video repetitions, the use of a CNN in such a framework, the reliance on purely unrealistic (i.e., not using realistic CG) synthetic data for training a computer vision system, changing the state of the system based on entropy, automatically working at multiple time scales, and more.

Our method is (i) robust to repetition variability in length and appearance (ii) online (iii) realtime (iv) starts and (v) stops counting automatically, supports (vi) multiple time scales and (vii) moving cameras. We are not aware of any other method that covers convincingly even one of these criteria, except (ii) in one previous Mocap publication.

The method is evaluated extensively on a very diverse set of benchmarks that were not used for training the system. In addition to many literature benchmarks, we provide a new one, which while being smaller than action recognition benchmarks, is much larger than all previous-work benchmarks used for this task combined. Evaluating systematically variants of our method allows us to quantify the relative contribution of each component. Lastly, our implementation is mature enough to interact freely with minimally instructed users in continuous live sessions.

As future work, we would like to handle multiple concurrent actions by generalizing the ROI mechanism, incorporating recent segmentation and proposal technologies. In addition, we would like to create long-ranged and hierarchical inferences across bouts of repeated actions based on the activations of the hidden layers.

References

[1] O. Azy and N. Ahuja. Segmentation of periodically moving objects. In *ICPR*, 2008. 2

[2] J. Barbič, A. Safonova, J.-Y. Pan, C. Faloutsos, J. K. Hodgins, and N. S. Pollard. Segmenting motion capture data into distinct behaviors. In *Graphics Interface Conf.*, 2004. 2

[3] A. Briassouli and N. Ahuja. Extraction and analysis of multiple periodic motions in video sequences. *PAMI*, 2007. 2, 6, 7

[4] C.-Y. Chen and K. Grauman. Watching unlabeled video helps learn new human actions from very few labeled snapshots. In *CVPR*, 2013. 2

[5] R. Cutler and L. Davis. Robust real-time periodic motion detection, analysis, and applications. *PAMI*, 2000. 2

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009. 4

[7] L. Fei-Fei and A. Karpathy. Stanford's cs231n class notes. http://cs231n.github.io/neural-networks-2/, Jan 26, 2015. 3

[8] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *AISTATS*, 2011. 4

[9] D. Gong, G. Medioni, S. Zhu, and X. Zhao. Kernelized temporal cut for online temporal segmentation and recognition. In *ECCV*. 2012. 2

[10] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. *PAMI*, 2007. 6, 7

[11] K. Grauman, G. Shakhnarovich, and T. Darrell. Inferring 3d structure with a statistical image-based shape model. In *CVPR*, 2003. 2

[12] S. Guimaraes, R. Coelho, and A. Torres. Counting of video clip repetitions using a modified bmh algorithm: Preliminary results. In *ICME*, 2006. 2

[13] O. Gunay, B. Toreyin, K. Kose, and A. Cetin. Entropy-functional-based online adaptive decision fusion framework with application to wildfire detection in video. *IEEE Trans. on Image Processing*, 2012. 5

[14] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 2006. 2

[15] M. Hoai, Z. Lan, and F. De la Torre. Joint segmentation and classification of human actions in video. In *CVPR*, 2011. 2

[16] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. *CoRR*, abs/1406.2227, 2014. 2

[17] M. Jain, J. van Gemert, H. Jegou, P. Bouthemy, and C. Snoek. Action localization with tubelets from motion. In *CVPR*, 2014. 4

[18] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *PAMI*, 2013. 2

[19] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 2

[20] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 2

[21] I. Laptev, S. Belongie, P. Perez, and J. Wills. Periodic motion detection and segmentation via approximate sequence alignment. In *ICCV*, 2005. 2

[22] Q. Le, W. Zou, S. Yeung, and A. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *CVPR*, 2011. 2

[23] G. Li, X. Han, W. Lin, and H. Wei. Periodic motion detection with roi-based similarity measure and extrema-based reference selection. *Consumer Electronics*, 2012. 2, 6, 7

[24] A. Lpez-mndez, J. Gall, J. Casas, and L. V. Gool. Metric learning from poses for temporal clustering of human motion. In *BMVC*, 2012. 2

[25] P. Matikainen, R. Sukthankar, and M. Hebert. Feature seeding for action recognition. In *ICCV*, 2011. 2

[26] S. Oh, J. Rehg, T. Balch, and F. Dellaert. Learning and inferring motion patterns using parametric segmental switching linear dynamic systems. *IJCV*, 2008. 2

[27] D. Oneata, J. Revaud, J. Verbeek, and C. Schmid. Spatio-temporal object detection proposals. In *ECCV*. 2014. 4

[28] L. Pishchulin, A. Jain, C. Wojek, M. Andriluka, T. Thormahlen, and B. Schiele. Learning people detection models from few training samples. In *CVPR*, 2011. 2

[29] E. Pogalin, A. Smeulders, and A. Thean. Visual quasi-periodicity. In *CVPR*, 2008. 2

[30] Y. Ren, B. Fan, W. Lin, X. Yang, H. Li, W. Li, and D. Liu. An efficient framework for analyzing periodical activities in sports videos. In *Image and Signal Processing*, 2011. 2, 6, 7

[31] E. Ribnick and N. Papanikolopoulos. 3D reconstruction of periodic motion from a single view. *IJCV*, 2010. 2

[32] E. Ribnick, R. Sivalingam, N. Papanikolopoulos, and K. Daniilidis. Reconstructing and analyzing periodic human motion from stationary monocular views. *CVIU*, 2012. 2

[33] B. Sarel and M. Irani. Separating transparent layers of repetitive dynamic behaviors. In *ICCV*, pages 26–32, 2005. 6, 7

[34] S. Satkin and M. Hebert. Modeling the temporal extent of actions. In *ECCV*, volume 6311. 2010. 2

[35] S. M. Seitz and C. R. Dyer. View-invariant analysis of cyclic motion. *IJCV*, 1997. 2

[36] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, 2011. 2

[37] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. *CoRR*, abs/1406.2199, 2014. 2

[38] Y. Sun, X. Wang, and X. Tang. Deep convolutional network cascade for facial point detection. In *CVPR*, 2013. 4

[39] Y. Sun, X. Wang, and X. Tang. Hybrid deep learning for face verification. In *ICCV*, 2013. 4

[40] X. Tong, L. Duan, C. Xu, Q. Tian, H. Lu, J. Wang, and J. Jin. Periodicity detection of local motion. In *ICME*, 2005. 2

[41] J. Yang, Z. Shi, and Z. Wu. Automatic recognition of construction worker activities using dense trajectories. In *International Symposium on Automation and Robotics in Construction and Mining*, 2015. 6, 7

[42] F. Zhou, F. D. la Torre, and J. K. Hodgins. Hierarchical aligned cluster analysis for temporal clustering of human motion. *PAMI*. 2